GLOBAL AUTOMATION SEARCH

A project report submitted to Rashtrasant Tukadoji Maharaj Nagpur University in partial fulfilment for the award of degree of

Bachelor of Engineering
In
Computer Science and Engineering

By

SURBHI KHATRI

Under the supervision of

Dr M. B. Chandak

Department of Computer Science and Engineering,
Shri Ramdeobaba College of Engineering And Management,
Nagpur-440013
(An Autonomous Institution of Rashtrasant Tukadoji Maharaj Nagpur University)

MAY 2020

CERTIFICATE

This is to certify that the Thesis on

"GLOBAL AUTOMATION SEARCH"

is a bona fide work by

SURBHI KHATRI

submitted to the Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur in partial fulfillment of the award of a Degree of Bachelor of Engineering, in Computer Science and Engineering. It has been carried out at the Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur during the academic year 2020-21.

Date : 1 MAY, 2020
Place: Nagpur

Nitin Bhojwani Industry Mentor VMware	Hariprasad Manchi Industry Mentor VMware
Dr M. B. Chandak Internal Guide/H.O.D Department of Computer Science and Engineering	Dr. Anupam Kher Dean Training & Placements

DECLARATION

I hereby declare that the project titled **"GLOBAL AUTOMATION SEARCH"** submitted herein, has been carried out in the Department of Computer Science and Engineering of Shri Ramdeobaba College of Engineering & Management, Nagpur. The work is original and has not been submitted earlier as a whole or part for the award of any degree /diploma at this or any other institution / University.

Date: 1 MAY 2020
Place: Nagpur

SURBHI KHATRI
ROLL NO. 18

Approval Sheet

This report entitled

"GLOBAL AUTOMATION SEARCH"

by

SURBHI KHATRI

is approved for the degree of Bachelor in Computer Science and Engineering.

Name & signature of Supervisor	Name & signature of External. Examiner
----------------------------	----------------------------
----------------------------	----------------------------
Name & signature RRC Members	Name & signature of HOD
----------------------------	----------------------------
----------------------------	----------------------------

Date: 1 MAY 2020

Place: Nagpur

ACKNOWLEDGEMENT

Every project small or big is not possible without the guidance and help of various numbers of people in various fields who have always given their valuable advice and support. We sincerely appreciate the inspiration, guidance, dedication and positivity shown by all those people who in some or the other way have contributed in making this project a success.

First of all, I am grateful to the Training and Placements Department of Shri Ramdeobaba College of Engineering and Management, Nagpur for successfully arranging the program for Internship for me. I also thank VMware for recruiting me as an intern and creating such a wonderful environment for learning and applying skills that are part of the Software and IT industry.

I am really fortunate that I had the kind association as well as supervision of my mentors Nitin Bhojwani and Hariprasad Manchi whose exemplary guidance, time and constant encouragement, stern hand and careful monitoring throughout the internship was so great that even my most profound gratitude is not enough.

I would also like to thank the entire SRE team, members of which were always available to help whenever we needed it. This heartfelt thanks also extends to everyone in my team who shared their experience, knowledge and way of working with all of us interns in this internship period.

At last but not the least, I want to thank all my professors and friends who have always been helping and encouraging me. I am indebted to our Head of Department, Dr M. B. Chandak who gave us a positive response and provided us various approaches through which the project could be implemented and developed further.

By –

SURBHI KHATRI 18

ABSTRACT

As searching of internal documents very frequently. Looking at current scenario the data is growing day by day which makes the searching slow. It is problematic now and going to be even more problematic in the future to search for the growing data. Slower results makes users impatient and also reduces productivity. Also to make it more user friendly, the search engine should have substring search and also incorporate natural language queries. So it is decided to handle this problems in two parts, first to index all the data we have for faster search and then applying NLP to get relevant results for natural language queries. Hence, we aim to make searching experience awesome by backing it with an appropriate search engine that deals with natural language queries.

For indexing the data, the project uses Elasticsearch which is a open source search and analytics engine and can store unlimited records free of cost. For NLP it uses Naïve Bayes algorithm and train the dataset using NLTK library for python which provides many complicated functionalities to be used directly. It uses Flask for backend and Angular for UI

Results: A flask project is created which has two major use cases first being to index all the data and provide autocomplete feature and second is to apply NLP for better search experience.

LIST OF FIGURES

INDEX

CHAPTER 1
INTRODUCTION

1.1 Background

The number of records are increasing rapidly. On average, more than 150 new records are created every day. To search in this growing data we require a powerful search engine which is faster and can handle natural language queries .

1.2 Problem Definition

To index all data for faster retrieval and provide substring search. Apply NLP to handle natural language queries.

1.3 Need of the system

The Project should lead to significant improvement in searching experience and can cope up with the growing data.

1.4 Objectives of the system

The objectives are as follows:

1. To index all records periodically.
2. To provide autocomplete search feature.
3. To create dataset from scratch according to our requirement and train NLP model.
4. To suggest relevant results according to user input.

CHAPTER 2
REVIEW OF LITERATURE

2.1 Definition of the system

As the data is increasing every day, a powerful search engine is needed for faster and user-friendly searching experience. Make searching experience better by backing it with an appropriate search engine that deals with natural language queries.

2.2 Features of the System

This is a domain specific search engine build on our own dataset created from internal documents. It uses Elasticsearch to store and index all the records, which is a open source search engine backed up by Lucene index. It will handle natural language queries as the data will be trained using a NLP model. It also provides substring search or autocomplete search feature.

2.3 Limitations of existing system

The existing system also has indexed data, but it's not scalable and it does not handle natural language queries properly. Users have to manually go through all the pages to find required information.

2.4 Advantages of current system

Searching is significantly faster in this system as we have indexed every record. I have used Elasticsearch which is scalable as there is no limit on the number of records that can stored. It is for made distributed systems, which makes parallel processing possible. We are using natural language processing to train a model which will able to provide hassle free searching functionality. The project is built using flask framework which is the best framework for web app development as it is has built-in web server and debugger. Below are few other advantages of this project:

1. It will reduce searching time and give relevant results without much efforts and it can cope up with growing data.
2. It provides substring search feature which make the system more user friendly.

CHAPTER 3
ANALYSIS AND DESIGN

In this chapter, the procedure for implementation of the project and a brief explanation of why it will be useful for implementing the proposed system is included, and a brief description of the current system development approach is counted.

3.1 Methodology

The project development methodology is as follows :

- Set up and configure Elasticsearch on a virtual machine .
- Create flask project and route it according to our requirement.
- Index all data.
- Enable autocomplete searching and search in all fields.
- Configure scheduler to automatically update any changes in data.
- Create dataset for training NLP model
- Train and test NLP model
- Add testing module.
- Create UI.
- Create CI/CD pipeline.

3.2 Use-cases

- **Substring search**

  It will provide fast searching with autocomplete feature and substring search. User will be able to search on all the fields as well as on a single field.
  - configure Elasticsearch and scale it.
  - Figure out what data you need to index
  - index it with proper structure and autocomplete analyzer
  - Write search queries for it.

- **Full-Text Search**
  - Create dataset for training NLP model
  - Train and test the model
  - Get predictions for natural language queries.

3.3 Flow of Control

- A background scheduler runs all the time and fetches new data from external services and send the results to Elasticsearch to index all new records.
- User interacts with UI and will be authenticated.
- If user is found to be authentic and he can enter the query otherwise he is redirected to error page.
- After entering the query, the user input is sent to Elasticsearch or NLP engine depending on the use case.
- The query is processed, and relevant suggestions are returned to the app.
- App then displays the results to user.

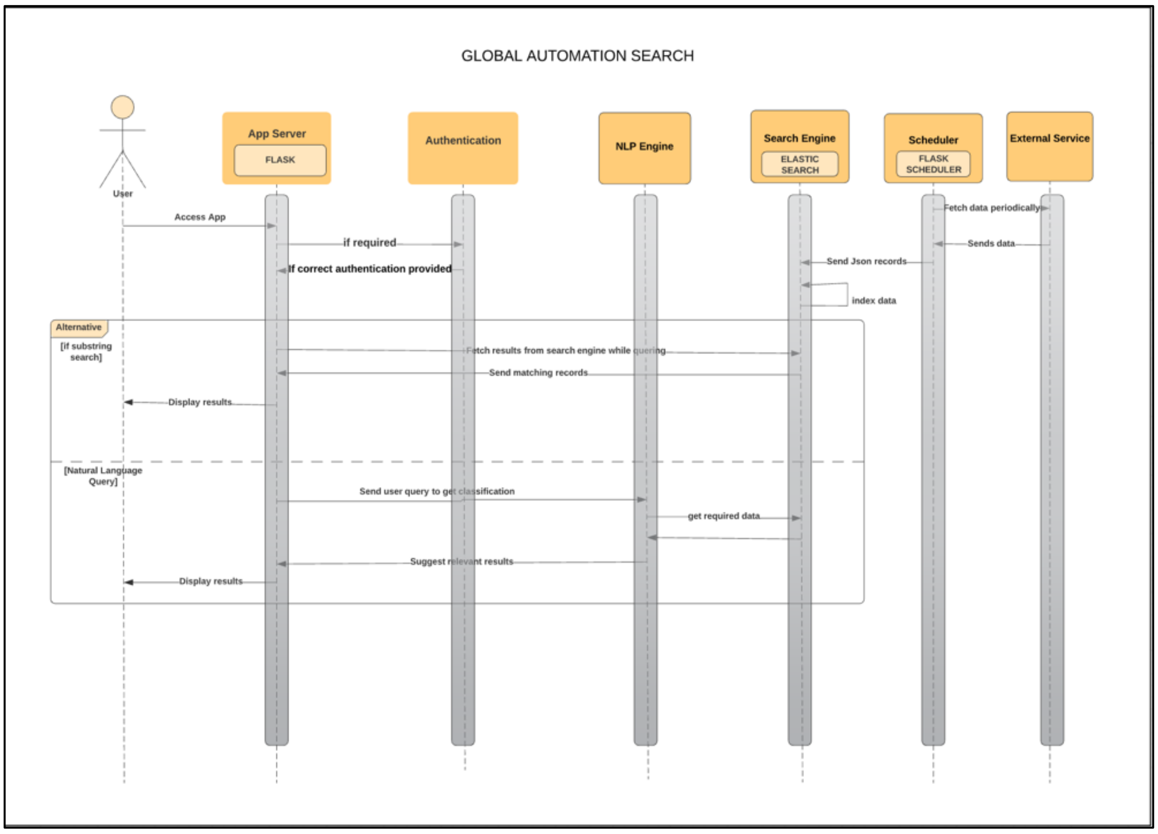


fig 3.1 Sequence Diagram

3.4 System Architecture

- The scheduler is triggered periodically get data from external services.
- New records are sent to Elasticsearch to index new data.
- The scheduler is also triggered periodically purge older data so that we don’t overload the server.
- Periodically retrain the NLP model with newer data to achieve higher accuracy.
- Apply search queries to have faster search results.

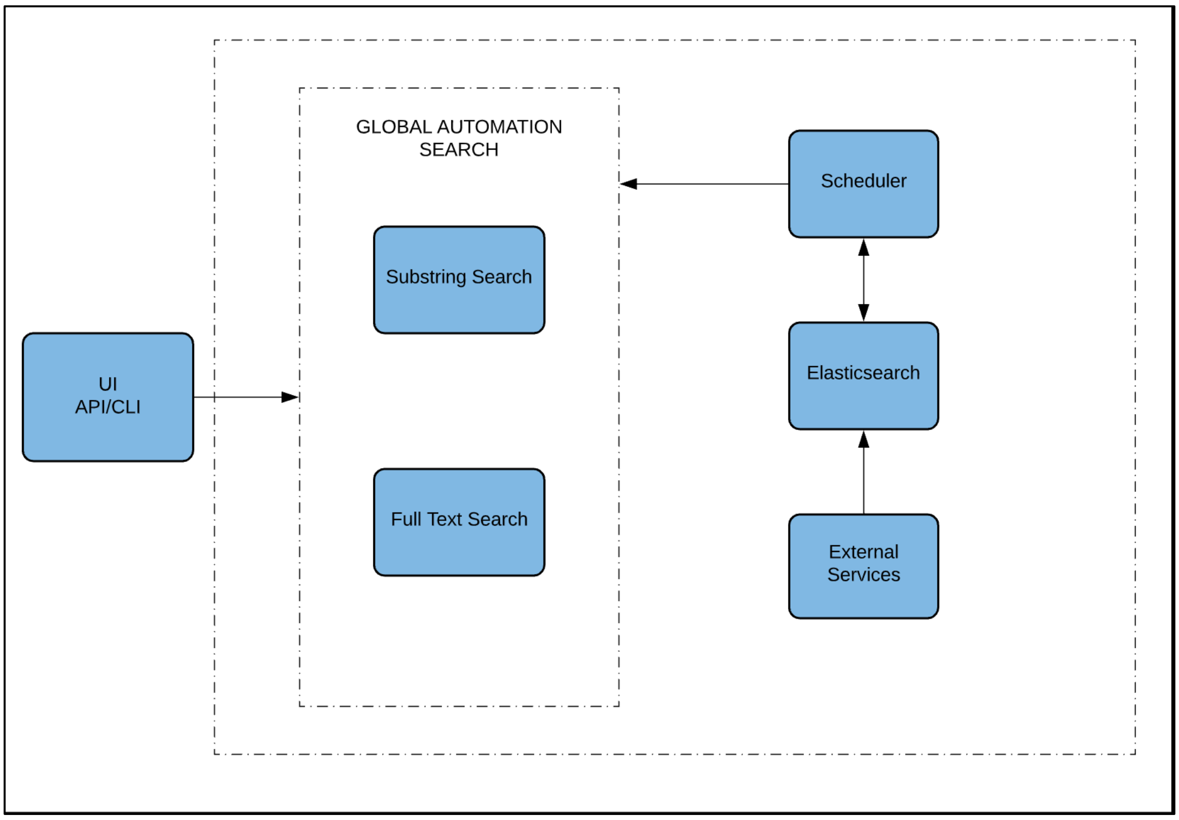


Figure 3.2 Architecture Diagram

CHAPTER 4
SYSTEM DESCRIPTION

1.1 Software tools

1. **Flask**
   a. Flask is a simple and minimalist web framework written in Python.
   b. Flask along with Bootstrap and SQLite can be used to easily develop full-functioning web apps.
   c. Among the important features of Flask are –
      i. built-in web server and debugger
      ii. unit testing support
      iii. RESTful request dispatching
      iv. Unicode support and good documentation
      v. Scheduler feature is available

2. **Python 3.7**

   Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

3. **Pandas**

   Pandas (software) In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

4. **NumPy**

   NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

5. **Beautiful Soup**

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

6. **Elasticsearch**

- Elasticsearch is a distributed, open source search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured.
- Elasticsearch can efficiently store and index it in a way that supports fast searches.

6.1 Elasticsearch Architecture

- A node is a single instance of Elasticsearch. It usually runs one instance per machine.
- Shards are individual instances of a Lucene index. Lucene is the underlying technology that Elasticsearch uses for extremely fast data retrieval.
- Elasticsearch is an abstraction that lets users leverage the power of a Lucene index in a distributed system

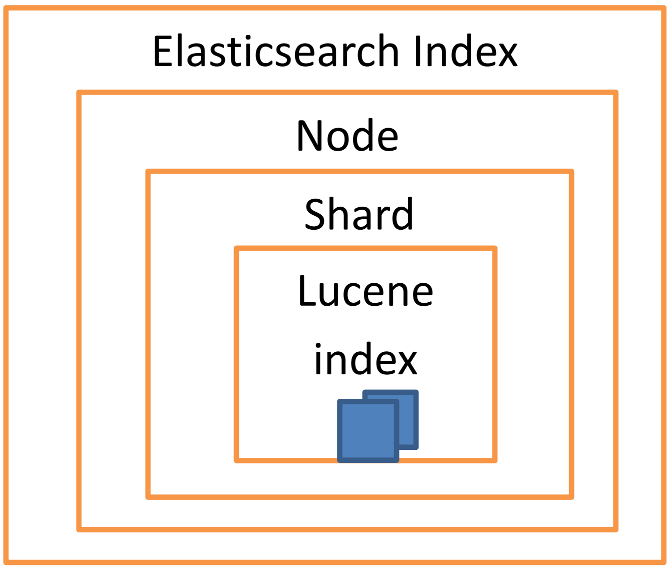


Figure 4.1 Elasticsearch Simplified Architecture

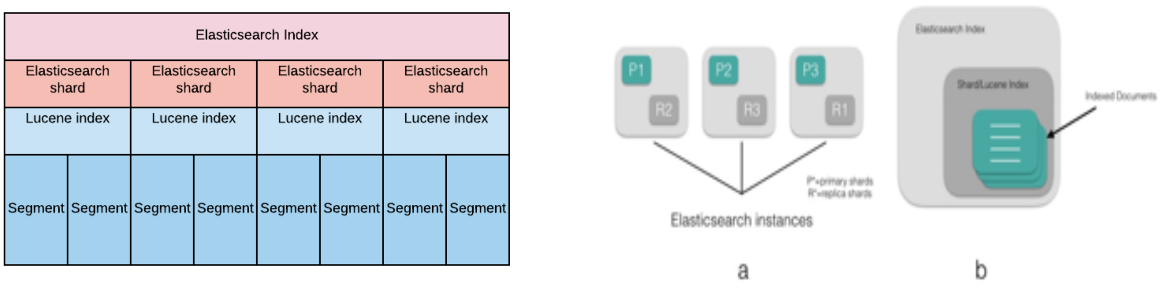
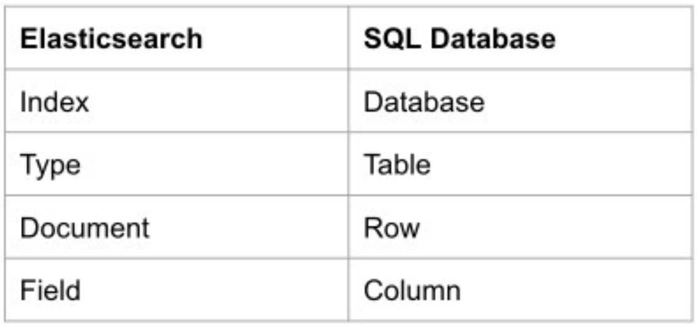


Figure 4.2 Elasticsearch Architecture



Elasticsearch	SQL Database
Index	Database
Type	Table
Document	Row
Field	Column

Figure 4.3 Analogy between Elasticsearch and RDBMS

6.2 Features and Benefits

- Scalable Map/Reduce model : Parallel processing on multiple shards.
- REST based : For flexible deployments by minimizing the number of ports needed to be open within a network.
- Self-contained : Small and efficient, at around 300KB and no extra dependencies.
- We do not have any limit on number of documents that can be indexed.
- We have replica shards for backup
- It follows distributed indexing.
- It has many types of searching and filtering techniques which make the search query more efficient.

CHAPTER 5
PROJECT IMPLEMENTATION

5.1 Elasticsearch in our UI

- The substring search feature will be integrated with our existing search bar.
- It will provide fast search results and is scalable as we have indexed our data.
- It will enable user to search every field for better precision.

5.2 Code for searching in Elasticsearch

```python
def search(index_name, query):

    out = {}
    query_list = []

    if validate_field(field):
        query_list = make_query(field, query)
    else:
        return "field not valid"


    result = es.search(
        index=index_name,
        body={'size': 10000,
              'query': {
                  'bool': {
                      'should': query_list
                  }
              }
              },
        scroll='1m'
    )

    for index, hit in enumerate(result['hits']['hits'], start=1):

        res = []
        for key in fields:
           res.append(key + " :" + (hit["_source"][key]))
        out[index] = res

    return out


def make_query(field, query):

    query_list = [{
        "regexp": {
            field: ".*" + query + ".*",
        }
    }]

    return query_list
```

CHAPTER 6
RESULT AND DISCUSSION

This model successfully provides autocomplete search feature. Elasticsearch accustoms very fast retrieval of data. It also provides great functionalities to store and index data and also make it scalable. The results are then displayed to user. For NLP the data will be sent to the NLP Engine and it will suggest all the relevant records. I understood how indexing works. I got a glimpse of Word2Vec a program to covert words to vector for use in Natural Language Processing. I learnt how to use NLTK for language processing purpose.

CHAPTER 7

FUTURE SCOPE

Though our model will be able to provide suggestions for most of the queries provided to the model but still it is not that accurate. With advancements in NLP these shortcomings can be compensated to a certain extent. Few of the goals for future are:

- Make use of NLP to provide natural language searching and improve its accuracy.
- Enhance the UI.
- Create a CI/CD pipeline for the project.

CHAPTER 8
CONCLUSION

A model to provide a better searching experience with autocomplete search feature which can cope up with growing data, is successfully created to suggest fast and precise results.